Python

Create a list

```
>>> # FIRST import numpy as np
...
>>> import numpy as np
>>> # CREATE a list
...
>>> data1 = [[1,2,3,4],[5,6,7,8]]
>>> data1
[[1, 2, 3, 4], [5, 6, 7, 8]]
>>> len(data1)
2
>>> data1.__class__
<class 'list'>
```

List is not having dimension, type, shape

```
>>> data1.ndim
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'list' object has no attribute 'ndim'
>>> data1.dtype
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'list' object has no attribute 'dtype'
>>> data1.shape
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'list' object has no attribute 'shape'
```

Create an array from list

```
>>> # Create and ARRAY from the list
...
>>> arr1 = np.array(data1)
>>> arr1
array([[1, 2, 3, 4],
       [5, 6, 7, 8]])
>>> len(arr1)
2
>>> arr1.__class__
<class 'numpy.ndarray'>
>>> arr1.ndim
2
>>> arr1.shape
(2, 4)
>>> arr1.dtype
dtype('int32')
>>>
```

Array of zeros

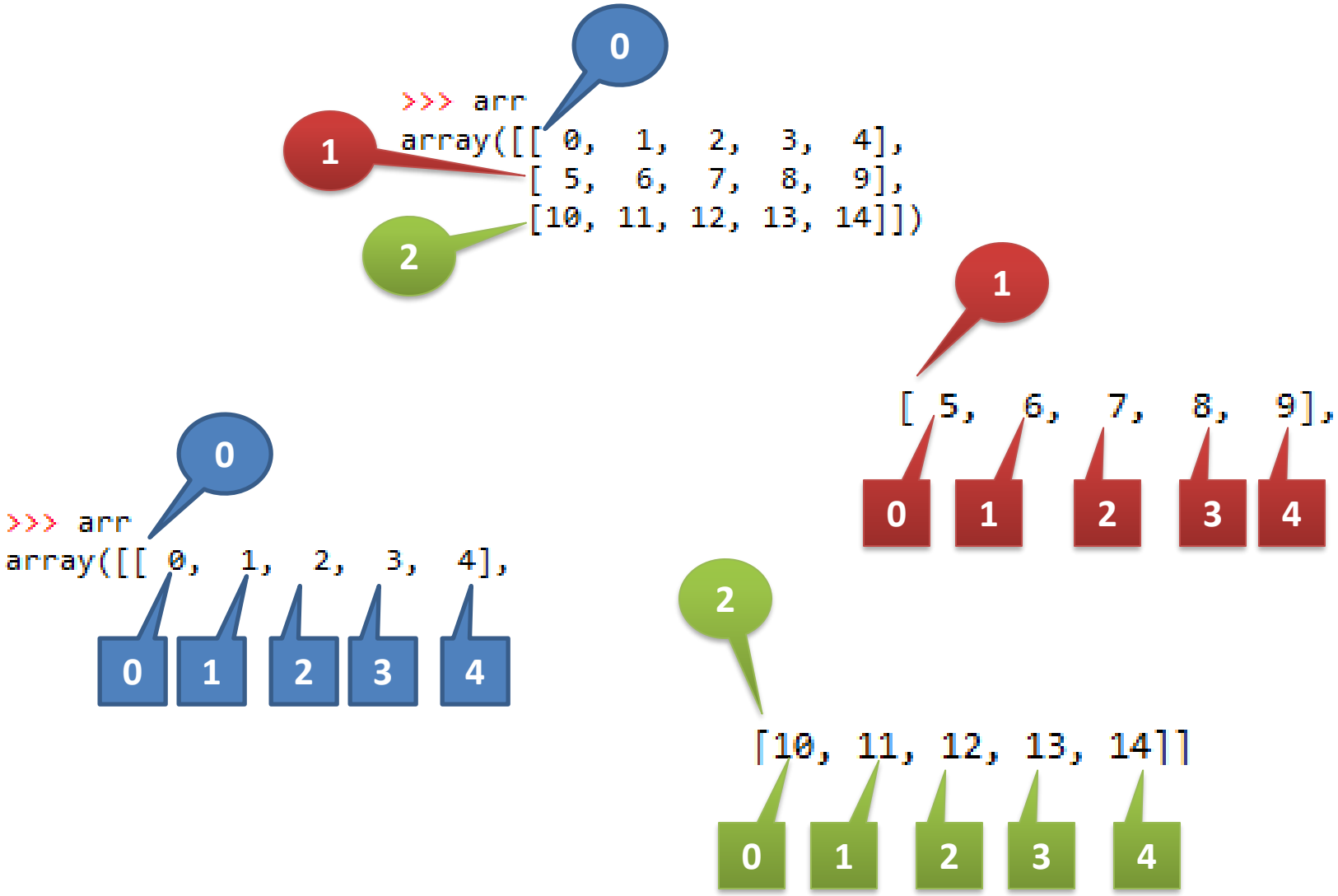
```
>>> # create an array of 0s
...
>>> np.zeros((3,6))
array([[ 0.,  0.,  0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  0.,  0.]])
>>>
```

Make an array of 3 by 5 dimension of first 15 numbers (including 0)

```
>>> k = np.arange(15)
>>> k
array([ 0,  1,  2, ..., 12, 13, 14])
>>> k.__class__
<class 'numpy.ndarray'>
>>> k.dtype
dtype('int32')
>>> isinstance(k, tuple)
False
>>> isinstance(k, list)
False
>>> k.reshape(3,5)
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14]])
```

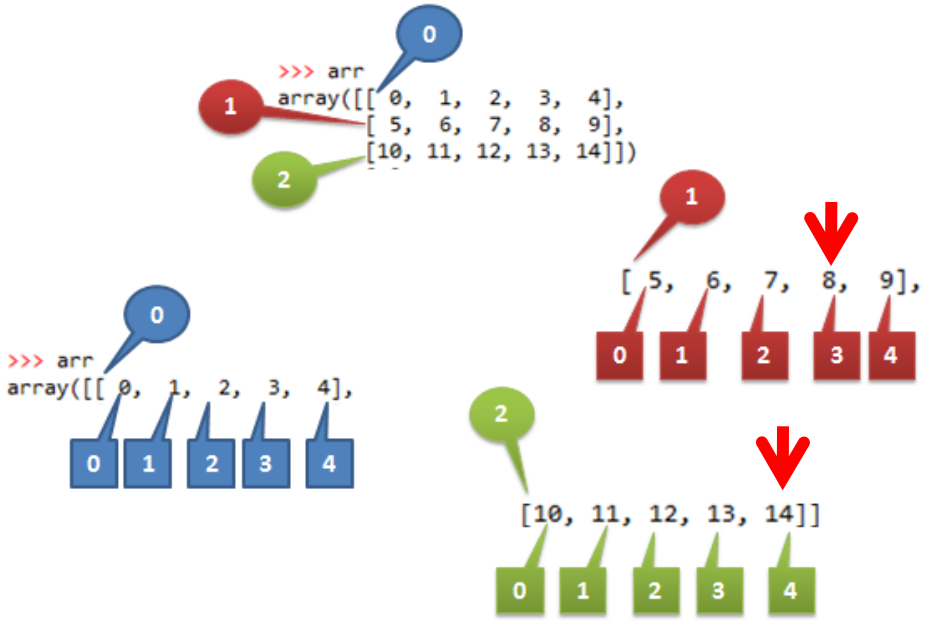
```
>>> len(k.reshape(3,5))
3
```

Indexing in arr

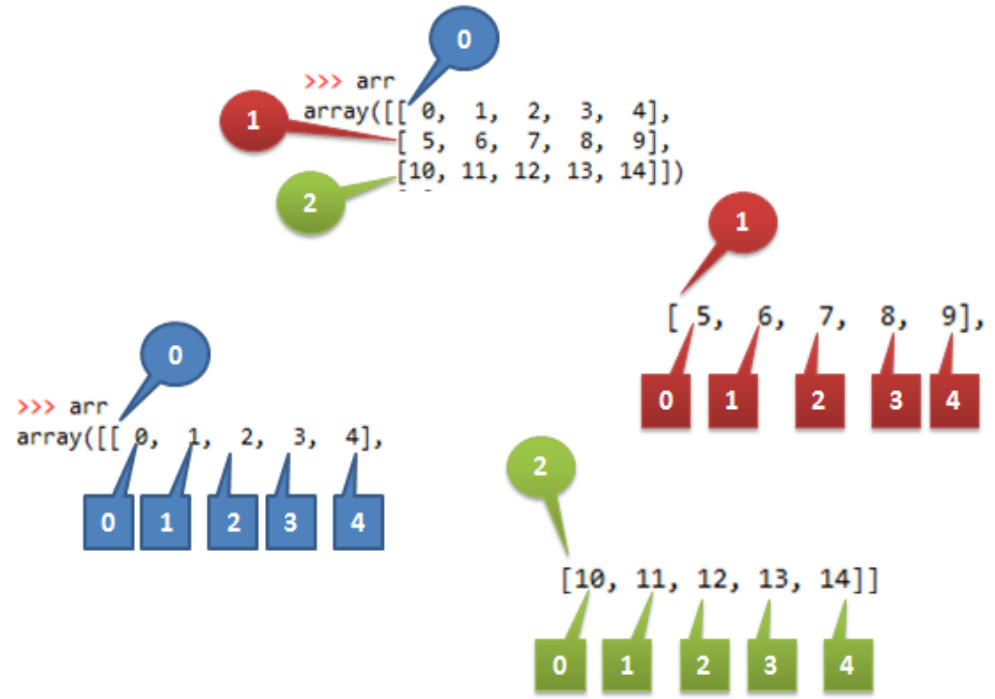


arr[2,4] and arr[1,3]

```
>>> arr = k.reshape(3,5)
>>> arr
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14]]
>>> arr[0]
array([0, 1, 2, 3, 4])
>>> arr[1]
array([5, 6, 7, 8, 9])
>>> arr[2]
array([10, 11, 12, 13, 14])
>>> arr[2,4]
14
>>> arr[1,3]
8
```



Array



2nd is ignored, 0 & 1 shown

```
>>> arr[:2]
array([[0, 1, 2, 3, 4],
       [5, 6, 7, 8, 9]])
>>> arr[2:]
array([[10, 11, 12, 13, 14]])
```

Array

```
>>> arr[:,0:2]
array([[ 0,  1],
       [ 5,  6],
       [10, 11]])
>>> arr[:,0]
array([ 0,  5, 10])
>>> arr[0:2, 0:2]
array([[0, 1],
       [5, 6]])
>>>
```

First TWO columns

First column

First TWO rows and first TWO columns

```
>>> arr
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14]])
```

Transpose of array

```
>>> arr.T
array([[ 0,  5, 10],
       [ 1,  6, 11],
       [ 2,  7, 12],
       [ 3,  8, 13],
       [ 4,  9, 14]])
```

```
>>> arr
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14]])
```

Square Root

```
>>> np.sqrt(arr)
array([[ 0.        ,  1.        ,  1.41421356,  1.73205081,  2.        ],
       [ 2.23606798,  2.44948974,  2.64575131,  2.82842712,  3.        ],
       [ 3.16227766,  3.31662479,  3.46410162,  3.60555128,  3.74165739]])
>>>
```

```
>>> arr
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14]])
```

Exponential of array

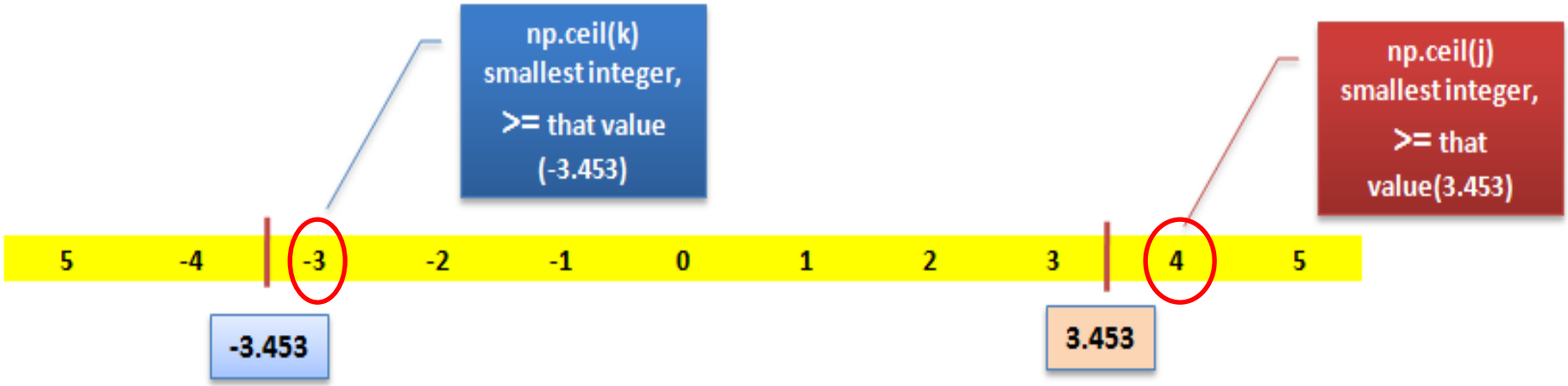
```
>>> np.exp(arr)
array([[  1.00000000e+00,   2.71828183e+00,   7.38905610e+00,
          2.00855369e+01,   5.45981500e+01],
       [  1.48413159e+02,   4.03428793e+02,   1.09663316e+03,
          2.98095799e+03,   8.10308393e+03],
       [  2.20264658e+04,   5.98741417e+04,   1.62754791e+05,
          4.42413392e+05,   1.20260428e+06]])
```

```
>>> arr
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14]])
```

ceil

```
>>> j = 3.453
>>> np.ceil(j)
4.0
>>> # Smallest integer, >,= the number
...
>>> k = -3.453
>>> np.ceil(k)
-3.0
```

```
>>> x = 3.4125
>>> x
3.4125
>>> y = round(x,2)
>>> y
3.41
```

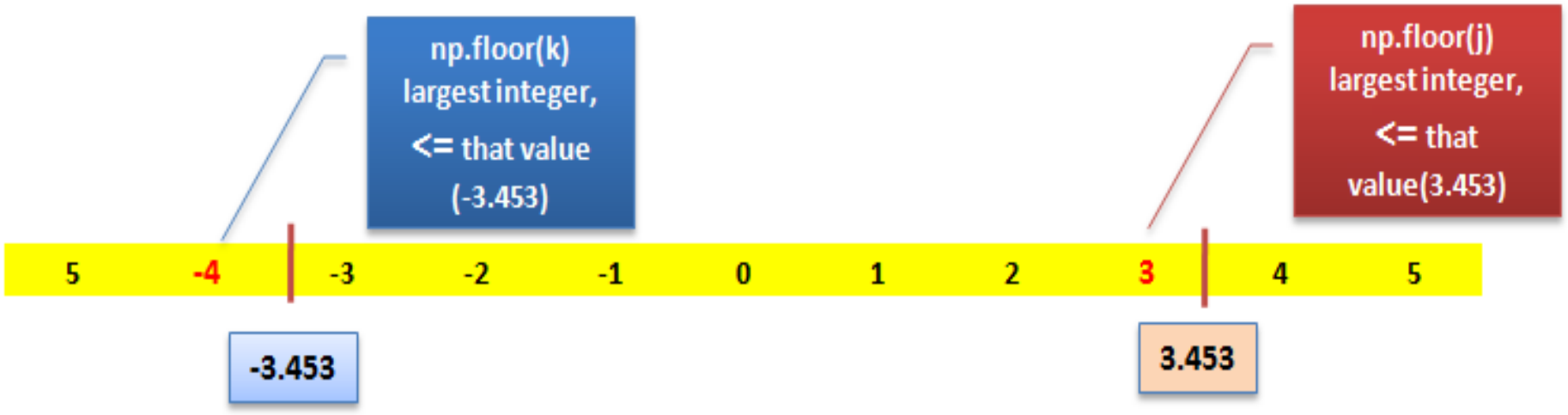


In ceil function, >= is important

floor

```
>>> np.floor(j)
3.0
>>> # Largest integer, <.= the number
...
>>> np.floor(k)
-4.0
```

```
>>> x = 3.4125
>>> x
3.4125
>>> y = round(x,2)
>>> y
3.41
```



In floor function, <= is important

np.greater_equal

```
>>> x1 = [4,2,1]
>>> x2 = [2,2,2]
>>> x1.__class__
<class 'list'>
>>> x2.__class__
<class 'list'>
>>> np.greater_equal(x1,x2)
array([ True,  True, False], dtype=bool)
>>>
```

Sum, mean, SD

```
>>> arr = np.array([[0,1,2], [3,4,5],[6,7,8]])
>>> arr
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
>>> arr.mean()
4.0
>>> np.mean(arr)
4.0
>>> arr.sum()
36
>>> np.sum(arr)
36
>>> arr.std()
2.581988897471611 2
>>> np.std(arr)
2.581988897471611 2
```